链滴

Tensorflow 进行图片分类

作者：itcalvin

原文链接：https://ld246.com/article/1575511381426

来源网站：链滴

Tensorflow进行图片分类

一、准备图片（最好分辨率一致，如果不一致，用以下脚本可处理成辨率一致）

```
#按照指定图像大小调整尺寸
def resize_image(image, height = IMAGE_SIZE_WIDTH, width = IMAGE_SIZE_HEIGHT):
    top, bottom, left, right = (0, 0, 0, 0)
    #获取图像尺寸
    h, w, _ = image.shape
    #对于长宽不相等的图片，找到最长的一边
    longest_edge = max(h, w)
    #计算短边需要增加多上像素宽度使其与长边等长
    if h < longest_edge:
        dh = longest_edge - h
        top = dh // 2
        bottom = dh - top
    elif w < longest_edge:
        dw = longest_edge - w
        left = dw // 2
        right = dw - left
    else:
        pass
    #RGB颜色
    BLACK = [0, 0, 0]
    #给图像增加边界，是图片长、宽等长，cv2.BORDER_CONSTANT指定边界颜色由value指定
    constant = cv2.copyMakeBorder(image, top , bottom, left, right, cv2.BORDER_CONSTANT,
alue = BLACK)
    #调整图像大小并返回
    return cv2.resize(constant, (height, width))
```

二、图片读取（转成tensorflow能识别的格式）

PS:代码中的 GHHZZ、GHJK、GHQSZ 等是图片所在文件夹的名称，我将它转成数字存储在数组中

```
#读取训练数据
images = []
labels = []
def read_path(path_name):
    for dir_item in os.listdir(path_name):
        #从初始路径开始叠加，合并成可识别的操作路径
        full_path = os.path.abspath(os.path.join(path_name, dir_item))

        if os.path.isdir(full_path):    #如果是文件夹，继续递归调用
            read_path(full_path)
        else:   #文件
            if dir_item.lower().endswith('.jpg'):
                # try:
                #     image = cv2.imread(full_path)
                #     image = resize_image(image, IMAGE_SIZE, IMAGE_SIZE)
                # except Exception as e:
```

```
                #     print('error_path:{0}'.format(full_path))
                #     break
                image = cv2.imread(full_path)
                image = resize_image(image, IMAGE_SIZE_WIDTH, IMAGE_SIZE_HEIGHT)
                #放开这个代码，可以看到resize_image()函数的实际调用效果
                #cv2.imwrite('1.jpg', image)
                images.append(image)
                #labels.append(path_name)
                if path_name.endswith('GHHZZ'):
                    labels.append(0)
                elif path_name.endswith('GHJK'):
                    labels.append(1)
                elif path_name.endswith('GHQSZ'):
                    labels.append(2)
                elif path_name.endswith('GHYBB'):
                    labels.append(3)
                elif path_name.endswith('GHYKB'):
                    labels.append(4)
                # elif path_name.endswith('GJJK'):
                #     labels.append(6)
                # elif path_name.endswith('GJMWB'):
                #     labels.append(7)
                # elif path_name.endswith('GJQSZ'):
                #     labels.append(8)
                # elif path_name.endswith('GJQYC'):
                #     labels.append(9)
                else:
                    labels.append(0)
    return images,labels


#从指定路径读取训练数据
def load_dataset(path_name):
    images,labels = read_path(path_name)
    #print(labels)
    #将输入的所有图片转成四维数组，尺寸为(图片数量*IMAGE_SIZE*IMAGE_SIZE*3)
    #图片为64 * 64像素,一个像素3个颜色值(RGB)
    images = np.array(images)
    print(images.shape)
    return images, labels

if __name__ == '__main__':
    if len(sys.argv) != 1:
        print("Usage:%s path_name\r\n" % (sys.argv[0]))
    else:
        images, labels = load_dataset("./data")
```

三、准备图片识别CNN模型

```
# coding = utf-8

import numpy as np
import tensorflow as tf
from tensorflow import keras
```

```
from load_data import load_dataset, resize_image, IMAGE_SIZE_WIDTH,IMAGE_SIZE_HEIGHT


#CNN网络模型类
class Model:
    def __init__(self):
        self.model = None

    #建立模型
    def build_model(self, dataset, nb_classes = 5):
        #构建一个空的网络模型，它是一个线性堆叠模型，各神经网络层会被顺序添加，专业名称为序模型或线性堆叠模型
        self.model =tf.keras.Sequential()

        # 添加神经网络层
        self.model.add(keras.layers.Convolution2D(filters=32, kernel_size=(3, 3),padding='same', activation='relu',
                                input_shape = dataset.input_shape))
        self.model.add(keras.layers.Convolution2D(filters=32, kernel_size=(3, 3), activation='relu')

        self.model.add(keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
        self.model.add(keras.layers.Dropout(0.25))
        self.model.add(keras.layers.Convolution2D(filters=64, kernel_size=(3, 3),padding='same', activation='relu'))
        self.model.add(keras.layers.Convolution2D(filters=64, kernel_size=(3, 3), activation='relu')

        self.model.add(keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
        self.model.add(keras.layers.Dropout(0.25))
        self.model.add(keras.layers.Flatten())
        self.model.add(keras.layers.Dense(512,activation='relu'))
        self.model.add(keras.layers.Dropout(0.5))
        self.model.add(keras.layers.Dense(nb_classes,activation='softmax'))


        #输出模型概况
        self.model.summary()

    #训练模型
    def train(self, dataset, batch_size = 32, nb_epoch = 20, data_augmentation = True):
        # sgd =keras.optimizers.SGD(lr = 0.01, decay = 1e-6,
        #          momentum = 0.9, nesterov = True) #采用SGD+momentum的优化器进行训练，先生成一个优化器对象
        adad= keras.optimizers.Adadelta(lr = 0.01,rho=0.95,epsilon=1e-08,decay = 1e-6)
        self.model.compile(loss='categorical_crossentropy',
                           optimizer=adad,
                           metrics=['accuracy'])  #完成实际的模型配置工作

        callbacks = [
            # Interrupt training if `val_loss` stops improving for over 2 epochs
            keras.callbacks.EarlyStopping(patience=2, monitor='val_loss'),
            # Write TensorBoard logs to `./logs` directory
            keras.callbacks.TensorBoard(log_dir='./logs')
            ]
```

```
        #不使用数据提升，所谓的提升就是从我们提供的训练数据中利用旋转、翻转、加噪声等方法创新的
        #训练数据，有意识的提升训练数据规模，增加模型训练量
        if not data_augmentation:
            self.model.fit(dataset.train_images,
                           dataset.train_labels,
                           batch_size = batch_size,
                           nb_epoch = nb_epoch,
                           validation_data = (dataset.valid_images, dataset.valid_labels),
                           shuffle = True,
                           callbacks=callbacks
                           )
        #使用实时数据提升
        else:
            #定义数据生成器用于数据提升，其返回一个生成器对象datagen，datagen每被调用一
            #次其生成一组数据（顺序生成），节省内存，其实就是python的数据生成器
            datagen =keras.preprocessing.image.ImageDataGenerator(
                featurewise_center = False,             #是否使输入数据去中心化（均值为0），
                samplewise_center  = False,             #是否使输入数据的每个样本均值为0
                featurewise_std_normalization = False,  #是否数据标准化（输入数据除以数据集的标准)
                samplewise_std_normalization  = False,  #是否将每个样本数据除以自身的标准差
                zca_whitening = False,                  #是否对输入数据施以ZCA白化
                rotation_range = 20,                    #数据提升时图片随机转动的角度(范围为0～180)
                width_shift_range  = 0.2,               #数据提升时图片水平偏移的幅度（单位为图片宽度占比，0~1之间的浮点数)
                height_shift_range = 0.2,               #同上，只不过这里是垂直
                horizontal_flip = True,                 #是否进行随机水平翻转
                vertical_flip = False)                  #是否进行随机垂直翻转

            #计算整个训练样本集的数量以用于特征值归一化、ZCA白化等处理
            datagen.fit(dataset.train_images)

            #利用生成器开始训练模型
            self.model.fit_generator(datagen.flow(dataset.train_images, dataset.train_labels,
                                                  batch_size = batch_size),
                                     steps_per_epoch = dataset.train_images.shape[0]/batch_size,
                                     epochs = nb_epoch,
                                     validation_data = (dataset.valid_images, dataset.valid_labels),
                                     callbacks=callbacks)

    MODEL_PATH = './calvin.model.h5'
    def save_model(self, file_path = MODEL_PATH):
        self.model.save(file_path)

    def load_model(self, file_path = MODEL_PATH):
        self.model = keras.models.load_model(file_path)

    def evaluate(self, dataset):
        score = self.model.evaluate(dataset.test_images, dataset.test_labels, verbose = 1)
        print("%s: %.2f%%" % (self.model.metrics_names[1], score[1] * 100))

    #
    def predict(self, image):
```

```
        #依然是根据后端系统确定维度顺序
        if  image.shape != (1, IMAGE_SIZE_WIDTH, IMAGE_SIZE_HEIGHT, 3):
            image = resize_image(image)
            image = image.reshape((1, IMAGE_SIZE_WIDTH, IMAGE_SIZE_HEIGHT, 3))

        #浮点并归一化
        image = image.astype('float32')
        image /= 255

        #给出输入属于各个类别的概率，我们是二值类别，则该函数会给出输入图像属于0和1的概率各
多少
        result = self.model.predict_proba(image)
        print('result:', result)

        #给出类别预测：0或者1
        result = self.model.predict_classes(image)

        #返回类别预测结果
        return result[0]
```

四、开启识别模式并随机取样验证数据集的准确性

```
import random
from load_data import load_dataset, resize_image, IMAGE_SIZE_WIDTH,IMAGE_SIZE_HEIGHT
from tf_dataset import Dataset
from tf_model import Model

if __name__ == '__main__':
    dataset = Dataset('./data/')
    dataset.load()
    model = Model()
    model.build_model(dataset)
    model.train(dataset)
    model.save_model(file_path = './model/calvin.model.h5')

if __name__ == '__main__':
    dataset = Dataset('./data/')
    dataset.load()
    #评估模型
    model = Model()
    model.load_model(file_path = './model/calvin.model.h5')
    model.evaluate(dataset)
```